MailAI: Revolutionizing Email Management with Artificial Intelligence *

Yosep Kim, Sumin Lim, Dohyeon Bu, and Misu Jung

Abstract. MailAI, hosted on AWS EC2, combines Django, MariaDB, JQuery, and Bootstrap technologies with Docker for efficient email management. It uses Let's Encrypt and Fernet Symmetric Encryption for security. Featuring AI-driven mail generation, analysis, and translation, and utilizing models like BERT, MailAI streamlines digital correspondence.

Keywords: Natural Language Processing (NLP) · Simple Mail Transfer Protocol (SMTP) · Internet Message Access Protocol (IMAP) · Cloud Computing · Artificial Intelligence (AI)

1 Introduction

- **Project Overview:** Integration of AI to enhance email management, featuring tools like summaries and keyword extraction for improved digital communication.
- **Motivation:** Addressing the challenge of efficiently managing large volumes of emails to save time and enhance productivity.
- **Goals and Proposal:** Simplifying email handling using AI technologies, focusing on efficient generation, analysis, and translation of emails.
- **Technical Approach:** Utilizing Django for web development, MariaDB for database management, BERT and KeyBERT for AI functionalities, and AWS EC2 for hosting. The design includes essential security measures like CSRF protection, encryption, and TLS.

The implementation of MailAI involves several key features:

- **Email Generation:** Utilizing RESTful API using Requests[7] and SMTP[6] for creating and sending emails.
- **Email Analysis:** Fetching emails via IMAP[3], storing them in MariaDB, and providing functionalities for email summaries, keyword extraction, and translations.
- **User Authentication and Security:** Implementing measures to ensure secure user access and data protection.
- **Database Management:** Robust management of user data and email content.

* Supported by IntueriAI.

2 Design for MailAI

2.1 Overall Architecture

The MailAI system is designed as a web-based application, leveraging a microservices architecture, components of this architecture include:

- **Web Browser**: The primary interface for the end-users.
- **AWS Route53**: Manages DNS services, directing user requests to the appropriate server.
- **Nginx Proxy**: Serves as the entry point in the EC2 instance, managing incoming traffic and forwarding it to the web server.
- **Docker Compose Network**: Hosts multiple containers including the web application, Nginx for SSL/TLS, and the database.
- **Django 4.2**: The main web framework for building the web application.
- **MariaDB**: The database for storing application data.
- **Persistent Volumes**: Used by Docker to persist database data.

Refer to Figure 1 for the system architecture diagram.

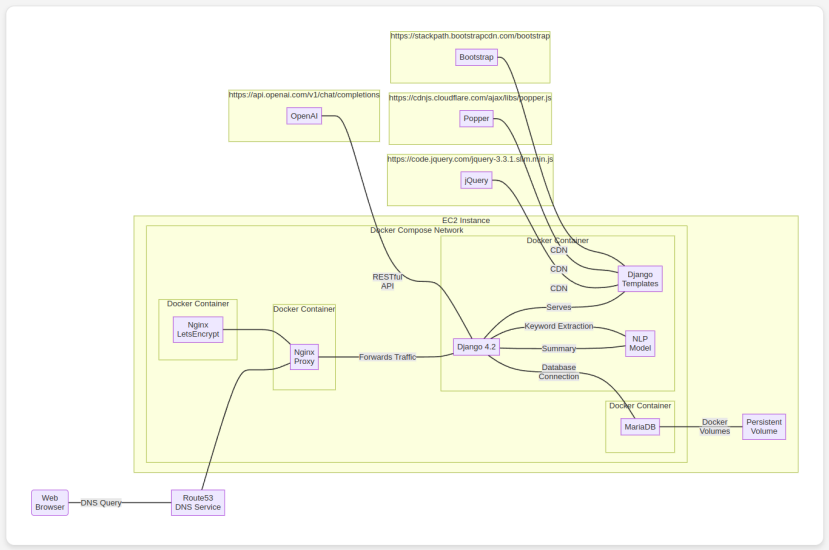

Fig. 1. The architecture of the MailAI system.

2.2 Design Rationale

The design choices for MailAI were driven by the need for a robust, scalable, and secure web application. Utilizing Django allowed for rapid development, while MariaDB offered a reliable database solution. The use of Docker ensured ease of deployment and consistency, and AWS services were chosen for their scalability and wide range of features.

2.3 Machine Learning Models and Algorithms

The NLP models used in MailAI are:

- **Transformer**: Google's Transformer, introduced in "Attention is all you need" [10], uses an encoder-decoder architecture with the Attention mechanism, effectively encoding relationships and importance in data [9].
- **BERT & KeyBERT**: Google's BERT [2], a pre-trained model, excels in bidirectional learning using the Masked Language Model (MLM). KeyBERT, an extension of BERT, is adept at keyword extraction using N-gram models, embedding vectors, and cosine similarity calculations, efficiently identifying relevant keywords.

The MailAI system leverages advanced NLP models from Hugging Face's model hub, configured as follows:

- **Summary Model**: Utilizes `bart-large-cnn-samsum` model of `philschmid` repository[8]for generating concise summaries of email conversations. This model, as per the Hugging Face model hub, excels in summarization tasks and has been fine-tuned for the SAMSum dataset. It's noteworthy that a newer version, `philschmid/flan-t5-base-samsum`, outperforms the BART version with improved ROGUE1 scores. This model is from facebook's Bart Large CNN Model [4]
- **Keyword Extraction Model**: Employs `all-mpnet-base-v2-embedding-all` of `LLukas22` repository [8] for extracting relevant keywords from emails. This model is fine-tuned on various datasets like squad, newsqa, and others, making it highly effective for feature extraction and sentence similarity tasks.

Listing 1.1 ensures dynamic model change across various NLP models, making MailAI a highly capable system in handling diverse systems.

```
bert:
  summary-model: "philschmid/bart-large-cnn-samsum"
  keyword-model: "LLukas22/all-mpnet-base-v2-embedding-
      all"
```

Listing 1.1. Model Configuration

3 Implementation

3.1 User Authentication

The user authentication process is streamlined and secure, comprising distinct functionalities for signing up, logging in, and account withdrawal.

- **Signup:** Users can sign up by clicking the Signup button on the Navbar or Home, as shown in Figure 2.

- **Login:** For existing users, logging in is facilitated via the Home Page or Navbar, demonstrated in Figure 3.
- **Logout:** For existing users, logging out is facilitated via the Home Page or Navbar.
- **Withdrawal:** Users can withdraw from the service via the Navbar, which requires password confirmation for security, depicted in Figure 4.

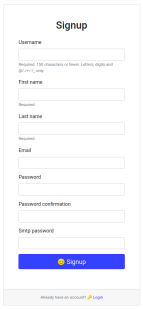

Fig. 2. Signup screen

Fig. 3. Login screen

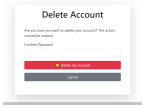

Fig. 4. Withdrawal screen

3.2 Generation

Generate Mail The mail generation process is user-friendly and intuitive, as detailed below. Users are required to fill out a form to generate an email, as shown in Figure 5. Subsequently, a send mail screen appears, indicating the Figure 6

Send Mail As Fiugre 6, Preview your mail after a short wait due to API response limits. Modify if needed and send. A loading screen appears during sending.

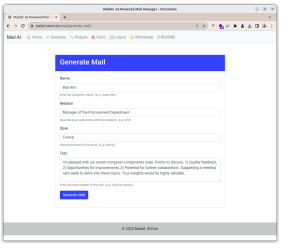

Fig. 5. Generate Mail screen

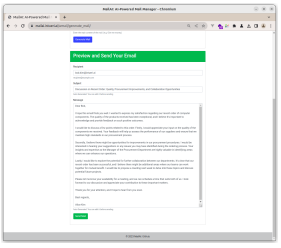

Fig. 6. Send Mail screen

3.3 Analysis

The user can access detailed email information by selecting an email from the email list window, as shown in Figure 7.

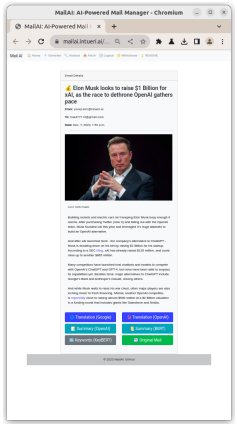

Fig. 7. Email Detail Screen

Fig. 8. Result of Translation

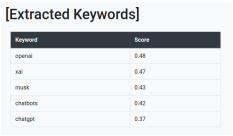

Fig. 9. Result of Keyword Extraction

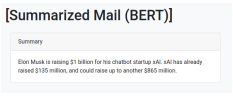

Fig. 10. Result of Summary

- **Translation:** The translation feature is illustrated in Figure 8. We utilize Google translator.
- **Keyword Extraction:** We use KeyBERT for keyword extraction (model: "LLukas22/all-mpnet-base-v2-embedding-all"), as depicted in Figure 9.
- **Summary:** For summarization, we employ the transformers model "philschmid/bart-large-cnn-samsum," shown in Figure 10.

3.4 Email Writing API

Generate Email

Endpoint: `/generate_mail/`
HTTP Method: GET, POST
Authentication: Requires login (apply the login_required decorator)

GET Request

Parameters: None
Response: Page providing a form for generating and sending mail

POST Request

Parameters:
- name (String): Recipient's name
- relation (String): Relationship with the recipient
- style (String): Mail style
- text (String): Mail body text
- generate_mail (Post flag): Clicked to generate mail button
- send_mail (Post flag): Clicked to send mail button

Response:
- Mail Generation Request (POST): Generate and provide initial values for mail body and subject in the form.
- Mail Sending Request (POST): Send the mail to the recipient, and redirect the page based on the success of the transmission. On success: `/generate_mail/?email_sent=True`. On failure: Display mail transmission and form validation error messages.

3.5 Email Analysis API

Show Email

Endpoint: `/show-emails/`
HTTP Method: GET
Authentication: Requires login (apply the login_required decorator)
Request Parameters:
- per_page (Number): Number of emails displayed per page (default: 10)
- page (Number): Current page number (default: 1)

Response: Page providing a list of emails based on the current page. Emails are sorted in descending order of the latest date.

Email Detail

Endpoint: `/emails/<int:email_id>/`
HTTP Method: GET, POST
Authentication: Requires login (apply the login_required decorator)

GET Request
Parameters: None
Response: Page providing detailed content of the email

POST Request
Parameters: modify (String): Type of email modification (translation, summary, bert_summary, keyword, llm_translation, original)
Response: Page providing the modified content of the email based on the selected modification type.

Handle Emails

Endpoint: `/handle-emails/`
HTTP Method: GET, POST
Authentication: Requires login (apply the login_required decorator)

GET Request
Parameters: None
Response: Page providing email handling options

POST Request
Parameters:
- fetch (Post flag): Clicked to fetch emails button
- reset (Post flag): Clicked to reset emails button

Response: When fetch is clicked: Attempt to fetch emails and redirect the page based on the result. When reset is clicked: Delete all emails for the current user and redirect the page.

3.6 Database

The Entity-Relationship Diagram in Figure 11 illustrates the database schema.

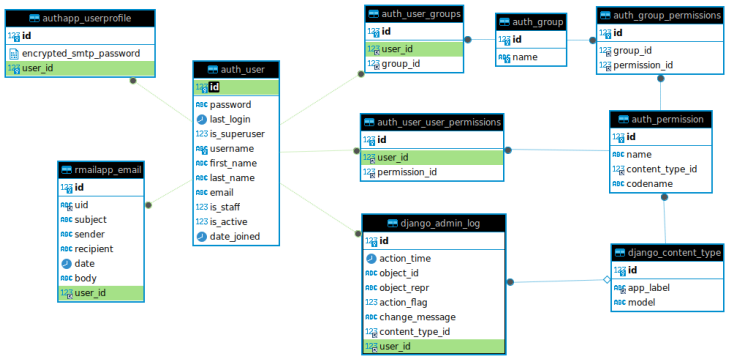

Fig. 11. Entity-Relationship Diagram of the MailAI Database.

3.7 Security

TLS We secure web applications using TLS for data encryption in transit, employing nginx as a reverse proxy. Nginx forwards HTTPS traffic, enabled by TLS, to the application server. Let's Encrypt manages certificate issuance and renewal.

Fernet Encryption Fernet, a symmetric encryption method provided by the cryptography library [1] in Python, is employed for securely handling sensitive data such as SMTP passwords. The Fernet encryption mechanism ensures that stored credentials remain inaccessible to unauthorized access.

4 Evaluation

4.1 Summary Feature

The summary feature uses the ‘bart-large-cnn-samsum‘ model by ‘philschmid‘ [8].

Model Specifications The model, trained on Amazon SageMaker using the Hugging Face container, offers efficient training and deployment. An enhanced version, ‘philschmid/flan-t5-base-samsum‘, provides improved performance.

Training Hyperparameters Table 1 shows significant hyperparameters used in training include

Table 1. Significant Hyperparameters used in Training

Hyperparameter	**Value**
Learning Rate	5×10^{-5}
Number of Training Epochs	3
Training and Evaluation Batch Size	4
Seed	7
FP16 Precision	Enabled

Performance Metrics The model demonstrates robust performance, as evidenced by its scores in various ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics.

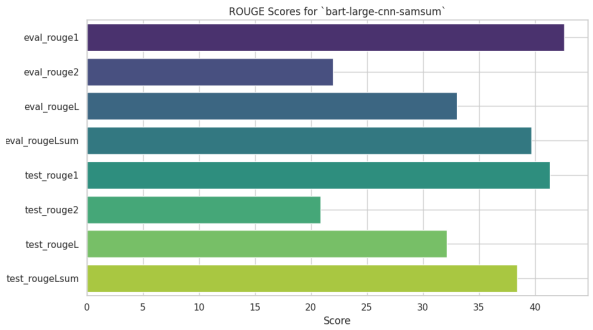

Fig. 12. ROUGE Scores of bart-large-cnn-samsum.

As shown in Figure 12, the model exhibits strong performance in both the evaluation and test phases, with high scores across all ROUGE metrics.

4.2 Keyword Extraction Feature

Our application's keyword extraction feature employs the 'all-mpnet-base-v2-embedding-all' model [5], an extension of MPNet, available on Hugging Face. Tailored for sentence similarity in English, this model is fine-tuned on diverse datasets like squad, newsqa, and various LLukas22 collections, providing a wide understanding of textual contexts.

Training Overview As detailed in Table 2.

Table 2. Key Hyperparameters in the Training Process

Hyperparameter	**Value**
Learning Rate	1×10^{0}
Per Device Batch Size	60
Effective Batch Size	180
Seed	42
Optimizer	AdamW
Weight Decay	2×10^{-2}
Number of Epochs	15
Mixed Precision Training	bf16

The model underwent 15 epochs of training, showing consistent improvement in loss reduction over time.

Evaluation Results The performance of the model was evaluated based on top-k accuracy metrics. The following table summarizes the evaluation results:

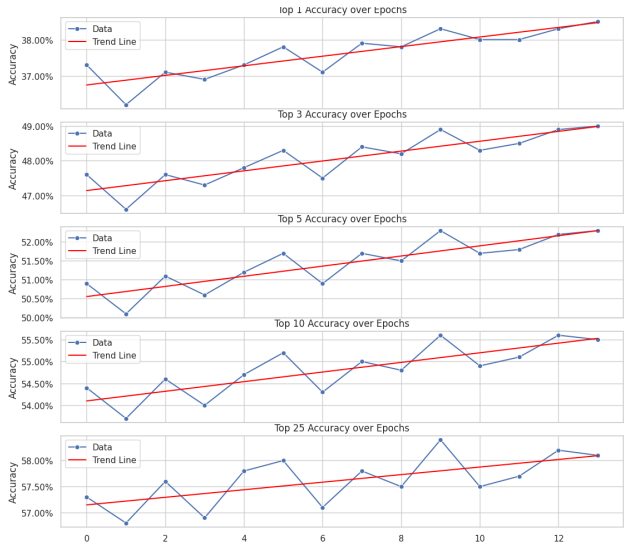

Fig. 13. Performance metrics of the model across epochs.

As Figure 13 illustrates, the model demonstrated consistent and reliable performance throughout the training and evaluation phases.

5 Limitations and Discussions

5.1 Challenges and Solutions

The development of MailAI involved addressing several key challenges:

- **Scalable Architecture:** Utilized AWS services for enhanced scalability and reliability.
- **Data Security:** Ensured secure data transmission using SSL/TLS through Nginx and further strengthened security with Letsencrypt and Django Cryptography.
- **Consistent Environment:** Implemented Docker for containerization, ensuring operational consistency across various environments.
- **Accurate NLP Functionalities:** Integrated advanced NLP models for higher accuracy in language processing tasks.
- **Insufficient Memory:** Addressed memory constraints by caching model data within the Docker Image for smoother operations.

- **Traffic Management:** Improved system responsiveness by switching Gunicorn's workers to Async Uvicorn, optimizing performance under high traffic.
- **Handling Image Data:** Dealt with email image data by reading through IMAP and displaying it in BASE64, ensuring seamless rendering.

5.2 Limits

MailAI's development and functionality are influenced by several inherent limitations:

- **Mail Body Size:** Limited to 2048 characters due to token size constraints and API costs.
- **BERT Token Limit:** The maximum size for the Summarizer is 1024 tokens, impacting the comprehensiveness of text summaries.
- **Response Time:** Variability in OpenAI API's response times may affect system responsiveness.
- **Browser Compatibility:** Requires a web browser supporting HTML5 for optimal performance.
- **Language Support:** Currently, the system only supports English, a limitation stemming from the model's capabilities.

These constraints are vital for understanding MailAI's current capabilities and areas for future development.

6 Related Work

This section compares MailAI with other AI-driven email tools, highlighting its unique features in the landscape of email management solutions.

6.1 YAMM and WriteMail.ai

YAMM (Your AI Mail Manager), available at `https://yamm.com/email/tool/ai-email-writer/`, offers AI features like Subject Line Generator, Email Writer, Sentiment Analyzer, and more. **WriteMail.ai** (`https://writemail.ai/`) focuses on simplifying email writing and ensuring privacy, with AI-assisted composition and reply generation, storing data in browser cookies.

6.2 Comparison with MailAI

MailAI differentiates itself from YAMM and WriteMail.ai in key areas:

- **IMAP Integration:** Offers direct IMAP integration for enhanced email management.
- **Model Flexibility:** Features configurable NLP models for adaptability (see Listing 1.1).
- **Privacy and Data Handling:** Maintains high standards in data security with advanced server-side handling.

MailAI stands out in the AI email tool landscape with its unique integration, model flexibility, and data security practices.

7 Conclusion

This document presents MailAI, an AI-enhanced email management system designed for both personal and business use. It integrates NLP models, IMAP, and a user-friendly interface to streamline digital communication.

MailAI's core features, including email generation, analysis, and translation, are powered by advanced technologies and hosted on AWS for scalability and security, incorporating TLS, Fernet encryption, and Docker containers.

Despite challenges in memory, traffic, image processing, and security, along with constraints like mail body size and response time, MailAI provides a robust solution, outperforming similar tools like YAMM and WriteMail.ai in IMAP integration, model adaptability, and data privacy.

As a significant advancement in email management technology, MailAI is poised to evolve and adapt to future user needs and technological developments.

References

1. cryptography.io: Fernet (symmetric encryption). https://cryptography.io/en/latest/fernet/ (2023), accessed: 2023-12-10
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019)
3. Kaukin, V.: imap-tools: Work with email by imap. https://pypi.org/project/imap-tools/ (2023), python version 3.5+
4. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. CoRR **abs/1910.13461** (2019), http://arxiv.org/abs/1910.13461
5. LLukas22: all-mpnet-base-v2-embedding-all. Hugging Face Model Hub (2021). https://doi.org/DOI (if available), https://huggingface.co/LLukas22/all-mpnet-base-v2-embedding-all, fine-tuned version of all-mpnet-base-v2 on various datasets. Used for sentence similarity and feature extraction.
6. Python Software Foundation: smtplib - SMTP protocol client. https://docs.python.org/3/library/smtplib.html (2023), documentation for the smtplib module in the Python standard library
7. Reitz, K.: requests: Python HTTP for Humans. https://pypi.org/project/requests/ (2023), a simple, yet elegant, HTTP library for Python
8. Schmid, P.: Bart large cnn samsum. Hugging Face Model Hub (2021). https://doi.org/DOI (if available), https://huggingface.co/philschmid/bart-large-cnn-samsum, model trained using Amazon SageMaker and Hugging Face Deep Learning container. Available at: https://huggingface.co/philschmid/bart-large-cnn-samsum
9. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks (2014)
10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf